# 目标

方案目标：分析剧场分镜功能，程序、策划、美术分工明确，程序制作插件分别提供给美术与策划使用自动生成数据与脚本提高制作效率。

# 其他游戏分析

# 犬夜叉项目剧情制作流程分析



## 剧情制作流程

1. 相机动画制作

相机动画由美术在场景编辑器制作，做好之后，策划把做好的animation动画直接导入到代码工程，与角色配合进行微调，调整完毕之后，在将相机动画资源打成ab包用于项目。

1. 接口设计

程序设计剧情中每个角色事件的接口，动作播放，特效播放，相机移动开关，资源加载，数据导出等等接口，提供给策划，策划编写lua脚本时直接调用这些接口完成剧情。

1. 剧情控制器制作

策划设置关键帧数据，每一个时间点发生什么事：如播放特效，切换人物动作等。每一个剧情副本对应一个这样的配置，保证剧情按照配置正常进行。每一个剧情配置需要策划不断运行游戏进行调整，完成之后将数据打包，在游戏运行时使用。

1. 相机控制脚本，策划在编写lua脚本时，同时在脚本中编写相机的控制逻辑，包括游戏相机和剧情相机的开关，剧情相机移动、抖动等动画的播放时机等
2. 复审

美术审核剧情效果，程序审核Lua脚本

优点：

1. 剧情中使用新模型而不用原来游戏中的模型可以在播放剧情时可以使用高模模型，给玩家带来更好的视觉感受
2. 该方案中的剧情逻辑与游戏逻辑相对独立，更易于维护。

缺点：

1. 剧情动画纯客户端表现，不需要与服务器通信.对于我们项目来说，播放技能就不能使用原来的技能流程，需要重新写剧情的动画逻辑。
2. 美术需要K相机动画帧，策划在调整时也会需要K动画帧，调整之后再给美术审核，开发效率低。
3. 策划需要编写Lua脚本，作为剧情控制器，每次调整都需要运行游戏观察效果，而且脚本完成需要程序复审lua脚本，降低开发效率。

问题：该方案主要问题在于需要策划编写Lua代码。第一，在编写初期很消耗时间，犬夜叉编写第一个剧情时间为一周；第二，策划编写的Lua代码与程序提供的接口相关联，出现问题时更不容易查找。

## 剧情动画播放流程。

1. 进入场景，加载场景资源时，将对应的剧情资源一并加载到游戏中。

function movie10000.Start(mgr, camera, func)

resMgr:LoadPrefab('camera\_1', reqItems, this.OnLoadCameraPrefab);

resMgr:LoadPrefab('10001', { "10001" }, this.LoadCharacterOK);

resMgr:LoadPrefab('18004', { "18004" }, this.LoadCharacterOK);

1. 触发剧情时，调用剧情管理器的对应接口，开始表演剧情

StoryManager.StartMovie();

由剧情管理器统一管理剧情动画的表演（相机移动，人物动作，以及播放特效等）

1. 推测犬夜叉项目剧情管理器的工作流程



# 倩女剧情流程分析

## 倩女剧情制作流程

1. 相机动画制作

相机动画有美术在场景编辑器制作，做好之后，策划把做好的animation动画直接导入到代码工程，与角色配合进行微调，调整完毕之后，再将相机动画资源打成ab包用于项目。

1. 接口设计

程序设计插件，给策划和美术提供制作效果的工具。

1. 剧情控制器制作

策划跳过编码过程直接使用工具制作剧情。

1. 复审

美术审核剧情效果。

以下是根据倩女的代码推演出的工作流程：



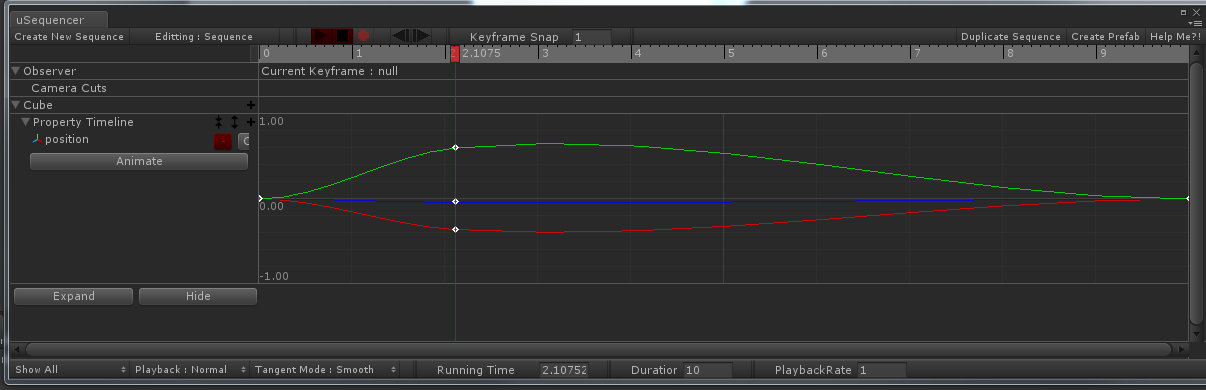
## 游戏中剧情逻辑

1. AmPlayer剧情的总控制器
2. AmTake单个剧情控制器
3. AmTrack时间轴控制器
4. AmKey每个关键帧对应的事件，时间等数据
5. AMBehavior 剧情中每个角色的事件控制器，每个角色相互独立互不影响



注意：倩女中的剧情资源也是剧情触发后加载的剧情相关的模型与游戏中的模型共存只是在剧情的相机中不渲染游戏里的模型

# uSequence插件分析：



优点：

1. 有友好的编辑界面，功能强大，可以制作相机动画，关键帧动画，控制Unity的动画控制器，控制Camera的开关
2. 所见即所得，在编辑旗下调整出来的剧情可以在编辑状态下随时调整，播放。
3. 开发效率高

缺点：

1. usequence插件会把Unity的animation数据转换为自己的帧数据，使用时会产生额外的内存开销。
2. 没有源码，它的数据结构和运行逻辑程序不可控（可以尝试修改它的反编译代码）

# 总结

根据以上分析，倩女幽魂的剧情制作流程开发效率更高，程序对代码的控制力更强，对于以后的功能维护更加有利。同时，可以让策划专注于剧情的设计，美术专注于效果的表现，程序专注于提供功能接口、工具和插件，分工更加明确，不会有功能上的穿插。因此我们倾向于使用倩女幽魂的剧情开发流程。

# 方向和问题

根据以上两个项目和usequence插件的分析，确定接下来的方向

1. 修改usquence反编译的代码，使用usquence的编辑界面能够极大地提高开发效率，而且更有利于美术制作效果。
2. 剧情控制部分仿照倩女的代码，每一帧对应多个事件，每个时间轴对应多个关键帧，每个剧情由多个时间轴组成，剧情总控制器控制每个剧情的播放。
3. 数据部分使用类似倩女的数据结构。

需要解决的问题：

1. 编辑器和资源的管理：

我们项目现在有四个编辑器：代码工程，角色技能编辑器，场景编辑器，UI编辑器，但是制作剧情是需要场景和角色相互配合，也就是要将剧情需要的角色特效动作导入到场景编辑器中进行编辑。如果想要以现在ab包的形式导入，则在编辑状态下无法看到效果。如果想要即时看到效果，就需要直接导入角色资源而不是ab包。（现在场景编辑器支持ab包的形式导入角色，必须在运行状态下才能看到效果）

1. uSequence反编译出的代码并不完整，需要修改它的控制逻辑，动画帧数据，以及资源管理逻辑代码
2. 多编辑器下的资源管理问题，剧情编辑插件是放在场景编辑器还是角色编辑器下
3. 场景动画的制作问题，如倩女中的地裂，是否需要新做场景。或者有其他方法。

# 需要做的测试

1. 测试Max导出的动画资源，是否需要在Unity中再次k帧移动位置。（不需要再次K帧，可以直接使用已测试）
2. 测试直接重做资放在Unity中作为剧情资源，资源大小是否可控（100M以内，未测试）
3. 测试使用角色编辑器的资源放到剧情编辑器中，k动作交互是否流畅，以及工作量的多少。（最终确定角色不使用角色编辑器的资源但是特效资源使用角色编辑器中的特效，如果有新添加的特效走原来的特效制作流程，但是剧情资源单独制作）

# 最终确定的流程

1. 剧情动画由美术在3DMax中制作完成。产出每个角色单独的fbx文件和相机轨迹的fbx文件，把角色的fbx文件导入到角色编辑器按照角色编辑器的制作流程打包资源，再把角色的ab包资源和相机轨迹的fbx文件导入到剧情编辑器的场景中，对相机进行微调，复现3DMax中的效果。（每个角色单独导出以便复用资源，角色模型，蒙皮，贴图等）(镜头在Unity中可调整)
2. 特效使用角色编辑器中的特效，把使用的特效导入到剧情编辑器，如果有添加或者修改则走之前的特效制作流程，在剧情编辑器中不修改，不上传特效资源,角色资源。
3. 在剧情编辑器中，程序为美术添加控制特效播放事件，对话播放事件的接口，供美术调整完善剧情效果。
4. 程序提供一件打包接口，美术在剧情编辑器中完成剧情编辑后，可以一键打包数据，将数据上传SVN，流程与场景编辑器类似。
5. 最终在游戏逻辑部分，由将剧情数据解析，将剧情复现到游戏中。

# 流程图

